Estudo Comparativo entre Métodos Ágeis e Tradicionais de Fabricação de Software

Trabalho de Conclusão de Curso

Engenharia da Computação

Bruno Junqueira Alves
Orientador: Prof. Sérgio Murilo Maciel Fernandes

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

Bruno Junqueira Alves

Estudo Comparativo entre Métodos Ágeis e Tradicionais de Fabricação de Software

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Agosto/2012.

De acordo

Recife

____/____________/_____

____

Orientador da Monografia

Dedico esse trabalho a todos que me deram o apoio necessário, minha família, meu orientador e amigo Sérgio Murilo e, principalmente, Deus.

Resumo

Nesse trabalho, foi feita uma pesquisa a respeito dos atuais métodos de fabricação de software utilizados pelas principais *software houses* do mundo. Também foram citados métodos usados no início da modernização dos processos de desenvolvimento. Através de entrevistas com especialistas na área, leitura especializada e um estudo de caso, foi possível chegar a uma comparação concisa entre os existentes métodos de construção de software.

Abstract

In this study, a survey was made about the current software development methods that are used by major software houses in the world. This work also mentions the methods used at the beginning of the modernization of development processes. Based on interviews with experts in the field, reading specialized books and performing a case study, it was possible to reach a concise comparison between the existing methods of software development.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A engenharia de software é uma disciplina relativamente nova. A ideia de 'engenharia de software' apareceu na década de 1960, exatamente em 1968. Nessa época, o hardware passava para um nível mais avançado de construção, tornando os computadores capazes de efetuarem tarefas antes impensáveis e tornando a fabricação mais barata. Mas, ao contrário dessa facilidade do hardware, o software estava ficando mais caro e a entrega com mais atraso. Portanto, novos métodos eram necessários para controlar a complexidade de sua fabricação.

Então foram criados os primeiros processos de fabricação de software, onde o objetivo era tornar a criação do software semelhante a um processo industrial, a fim de acompanhar o desenvolvimento do hardware das máquinas.

Houve um grande progresso desde essa época, mas ainda existem problemas para entregar os projetos no prazo e manter o custo inicialmente acordado. Por ser um processo intelectual, que usa julgamento humano, os processos de desenvolvimento de software são complexos e suas tentativas de automatização não obtiveram sucesso pleno.

Os primeiros métodos de desenvolvimento possuíam orientação estática, exatamente como uma linha de montagem. Com um tempo, as empresas começaram a adotar uma postura mais dinâmica, onde as mudanças no código não representam tantos problemas para a adaptação a um cliente específico. A partir daí, foram criados os métodos ágeis, cujo foco era nas pessoas e não nos processos estáticos de desenvolvimento.

1.1 Objetivo

Nesse trabalho, serão demonstrados os principais processos de desenvolvimento de software usados nas fábricas de software e também os que deixaram de ser utilizados devido a mudanças nos projetos do software. No final, será apresentada uma conclusão a partir de uma comparação entre esses processos.

1.2 Estrutura do trabalho

No capítulo 2, serão apresentados os métodos criados para o aperfeiçoamento do desenvolvimento do software desde o início da crise, durante a década de 1960.

Na primeira seção deste capítulo, a seção 2.1, será apresentada a metodologia em cascata. Na seção 2.2, serão apresentados os modelos incrementais de processo. Esses modelos se dividem em processo incremental e modelo RAD.

Na seção seguinte, 2.3, serão detalhados os modelos evolucionários de processo de software, nos quais serão citados o uso de prototipagem e o modelo espiral.

Na última seção do capítulo 2, será detalhado o desenvolvimento ágil. Nessa seção, será apresentada a sua história e porque esse modelo de processo de desenvolvimento vem sendo mais adotado pelas *software houses*.

No capítulo 3, será feita a comparação entre os modelos, citando suas vantagens e desvantagens em relação ao planejamento, execução e controle da fabricação.

No capítulo 4, será mostrado um estudo de caso a partir de uma empresa estabelecida no mercado, demonstrando a evolução e os ganhos da empresa ao utilizar a metodologia de desenvolvimento ágil em seu ambiente.

O capítulo 5, e último desse trabalho, levantará as conclusões e os trabalhos futuros possíveis a partir desse estudo comparativo, de modo que outros interessados realizem mais pesquisas em relação ao mercado de *software houses* brasileiro.

1.3 Metodologia

A metodologia adotada nesse estudo foi a leitura de livros, artigos e trabalhos especializados na área de desenvolvimento de software junto com entrevistas com profissionais que trabalham na área de desenvolvimento e gerência de qualidade de software.

Capítulo 2
Os processos de desenvolvimento de software.

O processo de desenvolvimento de software pode ser definido como uma coleção de padrões que definem um conjunto de atividades, ações, tarefas de trabalho, produtos de trabalho e/ou comportamentos relacionados e necessários ao desenvolvimento de software de computador. Pela combinação de padrões, uma equipe de software pode definir um processo que melhor satisfaça às necessidades de um projeto.

Nesse capítulo serão apresentados os principais métodos conhecidos de desenvolvimento de software, de modo que se possa ter um entendimento para a comparação desses métodos.

2.1 Modelo cascata

Existem ocasiões onde todos os requisitos de um sistema são bem compreendidos. O sistema a ser construído não precisa passar por muitas adaptações ou evoluções (aperfeiçoamentos), tornando sua implementação linear.

O modelo cascata, ou clássico, foi a primeira metodologia de desenvolvimento de software apresentada. Ela usa uma abordagem sistemática e sequencial para o desenvolvimento do software [4]. É um modelo onde a etapa seguinte só é executada após a anterior ser totalmente finalizada. Para ser iniciada uma etapa, a anterior precisar estar totalmente documentada e aprovada. Essa metodologia surgiu em um contexto de desenvolvimento de software muito diferente do atual, baseado apenas em um *mainframe* e terminais burros [6]. A Figura 1 ilustra bem o processo.
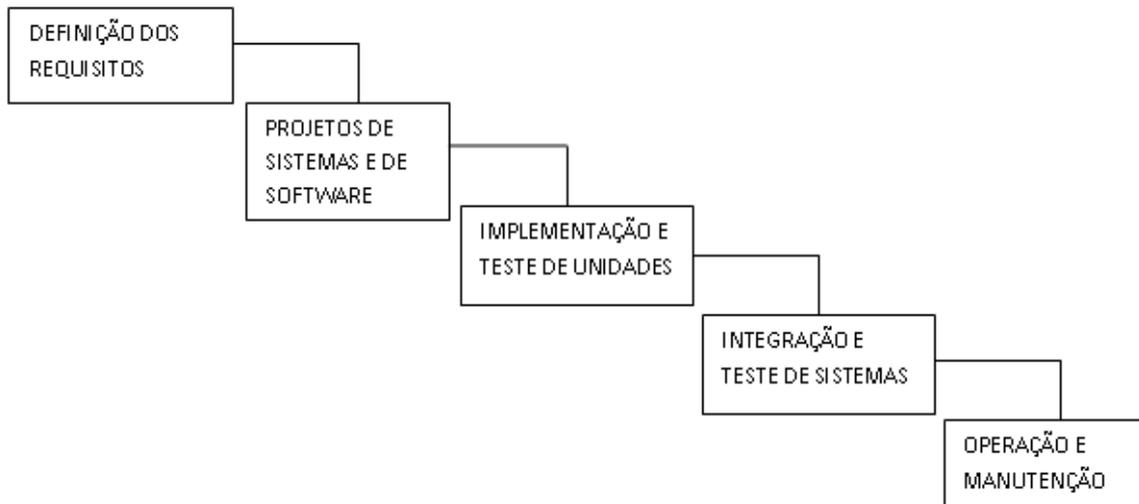
Figura 1. Etapas da metodologia cascata
[Fonte: retirado de [6]]

1. **Definição de requisitos**: As funções, as restrições e os objetivos do sistema são estabelecidos através de entrevista com usuários do sistema. Após isso, esses três pontos são definidos em detalhes e servem como uma especificação do sistema.

2. **Projetos de sistemas e de software**: O processo de projeto de sistemas agrupa os requisitos em sistemas de hardware e software [9]. Ele estabelece uma arquitetura do sistema geral. O projeto de software envolve a identificação e a descrição das funções fundamentais do sistema de software e suas relações.

3. **Implementação e teste de unidades**: Durante esse estágio, o projeto de software é compreendido como um conjunto de programas ou unidades de programa [4][7]. O teste de unidades certifica que as especificações de cada unidade foram atendidas.

4. **Integração e teste de sistemas**: As unidades de programa ou programas individuais são integrados e testados como um sistema único a fim de garantir que todas as especificações foram atendidas. Depois do teste de sistemas, o software é entregue ao cliente.

5. **Operação e manutenção**: Normalmente, embora não necessariamente, esta é a fase mais longa do ciclo de vida [7]. O sistema é instalado e sua operação é iniciada. A manutenção envolve corrigir erros que não foram descobertos anteriormente, em estágios anteriores do ciclo de vida. Com isso, as implementações do sistema são melhoradas e a quantidade de funções é aumentada à medida que novas especificações são descobertas.

O maior problema dessa metodologia é a sua alta burocracia e inflexibilidade na divisão do processo em fases bem distintas [6]. Esses dois fatores levam a um alto número de atrasos ou até a cancelamentos de projetos. Em relação aos atrasos, os custos se elevam, superando os custos previstos inicialmente, tornando o software mais caro.

Outro problema originado a partir desses dois pontos é o da qualidade do software. Os que são entregues dentro do prazo, e custando o acordado inicialmente, possuem qualidade duvidosa, pois houve muita pressão nos desenvolvedores para cumprir os prazos.

Hoje em dia, o trabalho em software é de ritmo rápido e com grande quantidade de pedidos de modificações. Com isso, o modelo cascata é inadequado para esse tipo de trabalho. O modelo pode ser útil em projetos onde os requisitos são fixos e o trabalho deve seguir de modo linear.

2.2 Modelos incrementais de processo

Há situações em que os requisitos iniciais do software são razoavelmente definidos, mas o escopo global do esforço de desenvolvimento elimina um processo puramente linear. Além disso, pode haver a necessidade de fornecer um conjunto limitado de funcionalidades do software aos usuários e depois refinar e expandir aquela funcionalidade em versões posteriores do software. Em tais casos, o processo de desenvolver o software incrementalmente é escolhido [4].

2.2.1 O modelo Incremental

O modelo incremental possui elementos do modelo cascata, que é aplicado de maneira iterativa [4]. No modelo incremental, são aplicadas sequências lineares de uma forma racional à medida que o tempo passa, como mostrado na Figura 2. Cada sequência linear produz "incrementos" do software passíveis de serem entregues.
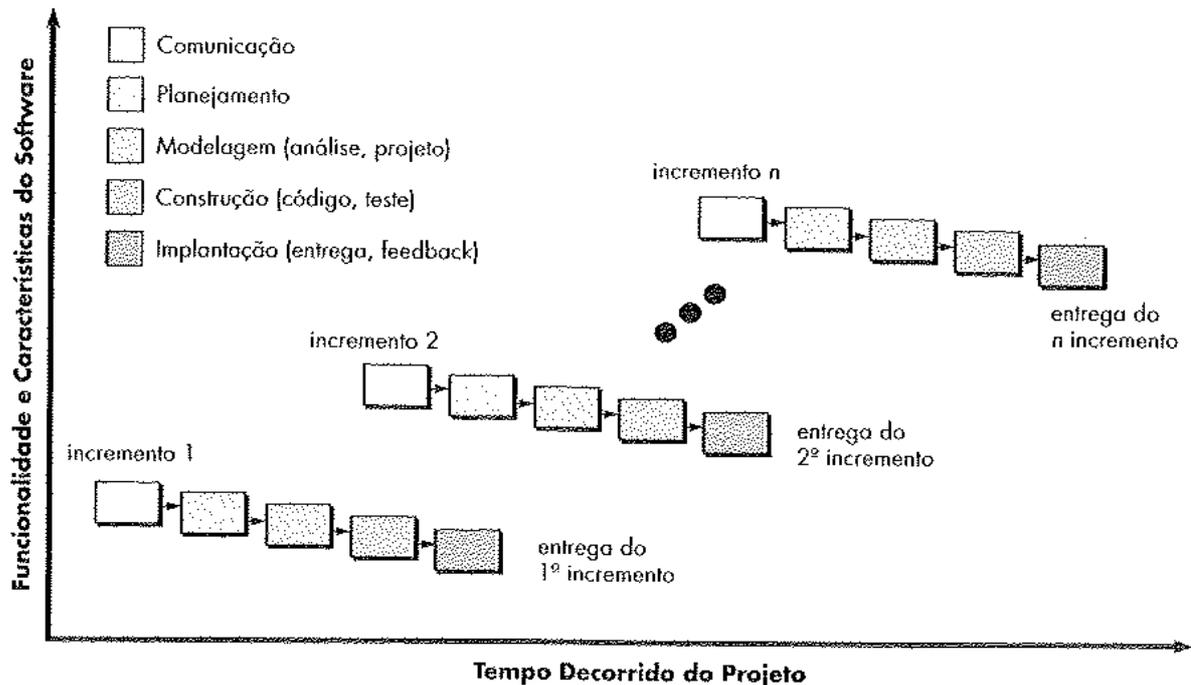
Figura 2. Modelo Incremental

[Fonte: retirado de [4]]

No processo incremental, o primeiro incremento desenvolvido é chamado de núcleo. Esse núcleo contém as características mais básicas do software. Ao longo do tempo o usuário solicita melhorias, ou seja, novos incrementos. São feitos novos planos de projeto a fim de que o software seja aperfeiçoado. Esse processo é repetido até que o sistema esteja com todas as funcionalidades implementadas.

A vantagem do uso desse modelo é a previsão de uso de mão de obra extra e de novas tecnologias de hardware. Através desse modelo, pode-se prever o uso de mais ou menos desenvolvedores. No caso de novas tecnologias de hardware, podem ser feitos planos de projeto de modo que se evite o uso dessas novas tecnologias com data de lançamento incerta.

O modelo incremental é iterativo por natureza [4]. E tem o objetivo de apresentar um novo produto operacional a cada incremento.

2.2.1.1 RUP (*Rational Unified Process*)

O RUP é um processo proprietário de engenharia de software. Foi criada pela Rational Software Corporation, adquirida posteriormente pela IBM, ficando conhecida como RUP IBM ou IRUP.

Esse processo fornece técnicas às equipes de desenvolvimento de software objetivando o aumento da produtividade seguindo uma abordagem prescritiva. O RUP se baseia no paradigma de Orientação a Objetos e é documentado utilizando UML (*Unified Modeling Language*) para ilustrar os processos em ação.

O principal objetivo do RUP é atender as necessidades dos usuários garantindo uma produção de software de alta qualidade que cumpra um cronograma e um orçamento previsíveis [8].

É considerado um processo pesado (tradicional) por muitos especialistas e é preferencialmente aplicável a grandes equipes de desenvolvimento e a grandes projetos.

O RUP permite definir quem é o responsável pelo o que deve ser feito, como deve ser feito e quando deve ser feito, descrevendo todas as metas específicas de desenvolvimento para que sejam alcançadas. A Figura 3 ilustra de forma geral esse processo.
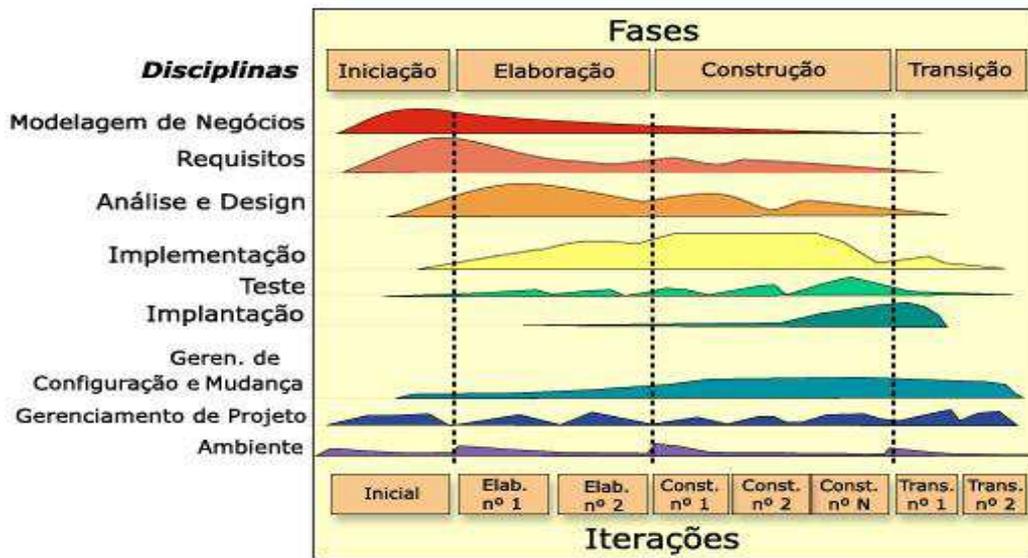
Figura 3. Visão geral do RUP

[Fonte: retirado de [9]]

O RUP organiza o desenvolvimento em quatro fases, como ilustrado na figura 4 e explicado a seguir. Cada fase tem um papel fundamental para que o objetivo seja cumprido:

- **Concepção ou Iniciação**: Essa fase abrange as tarefas de comunicação com o cliente e planejamento. Ocorre a identificação e a especificação dos casos de uso do projeto. Também é feito um plano de projeto, analisando seus riscos, delimitando os custos e os prazos e estabelecendo as prioridades.

- **Elaboração**: É realizada a análise da extensão do sistema. Também é definida a arquitetura que será utilizada no sistema. Essa arquitetura deve ser estável e robusta. Usando essa arquitetura como guia, é feita a análise e o projeto dos casos de uso, posteriormente documentados.

- **Construção**: O objetivo dessa fase é o desenvolvimento de componentes e outros recursos do sistema. É nessa fase que ocorre a produção de códigos e os testes *alfa* para certificar que os casos de uso estão sendo implementados corretamente. Devem-se aplicar processos de testes estáveis.

- **Transição**: O foco nessa fase é assegurar que o software esteja disponível para seus usuários finais [9]. As atividades nessa fase incluem o treinamento dos usuários finais e a realização dos testes *beta* do sistema visando que o mesmo tenha os níveis de qualidade adequados.

Figura 4. Fases de desenvolvimento do RUP

[Fonte: retirado de [8]]

O RUP tenta diminuir os riscos do desenvolvimento, a fim de torná-lo mais eficiente, utilizando seis práticas básicas a serem executadas durante o projeto:

- **Desenvolver iterativamente** – Esta prática permite que requisitos sejam identificados ou modificados mais facilmente, ocorra uma integração progressiva de elementos ao software, ocasionando uma melhora na identificação dos riscos.

- **Gerenciar requerimentos** – Provê uma maneira prática de produzir, organizar e comunicar os requisitos de um projeto. O gerenciamento de recursos acarreta um melhor controle sobre projetos complexos [10].

- **Utilizar arquiteturas baseadas em componentes** – O processo foca no desenvolvimento inicial de uma arquitetura robusta, antes do comprometimento de recursos em larga escala. Essa prática descreve como projetar uma arquitetura flexível, que suporta mudanças e promove

um reuso de software efetivo. O RUP provê uma aproximação sistemática para definir uma arquitetura através de componentes novos ou existentes [10]. Um componente pode ser definido como um módulo não-trivial ou um subsistema que cumpre uma função clara.

- **Modelar visualmente** – O processo mostra como modelar visualmente um software para identificar a estrutura e o comportamento da arquitetura e dos componentes. Isso permite entender melhor não só a concepção e complexidade do sistema, como também dimensionar, além de facilitar a identificação e a solução de problemas. O RUP, através da UML, permite uma modelagem compreensiva de sistemas complexos, formando uma base para a implementação e capturando requisitos com precisão.

- **Verificação contínua da qualidade** – O RUP toma a qualidade como responsabilidade de todos os integrantes do projeto. Durante o ciclo de vida do projeto, os integrantes implementam, medem e avaliam tanto a qualidade do processo como a do produto.

- **Controle de mudanças** – O RUP mostra como é possível controlar, monitorar e identificar mudanças para obter um desenvolvimento iterativo com sucesso. Isso ajuda o software a obter uma boa qualidade durante e depois de seu desenvolvimento.

2.2.2 O modelo RAD

O RAD (*Rapid Application Development,* desenvolvimento rápido de aplicação) é um modelo de processo focado no curto desenvolvimento. O modelo RAD é uma adaptação “de alta velocidade” do modelo cascata, no qual o desenvolvimento rápido é alcançado com o uso de uma abordagem de construção baseada em componentes [4].

Esse modelo é altamente recomendado para casos de projeto onde as principais funcionalidades podem ser implementadas em menos de três meses. Cada função pode ser tratada por uma equipe RAD diferente, no final juntando todas

as funções em um todo. A Figura 5 ilustra bem como acontece um projeto usando metodologia RAD em seu desenvolvimento.
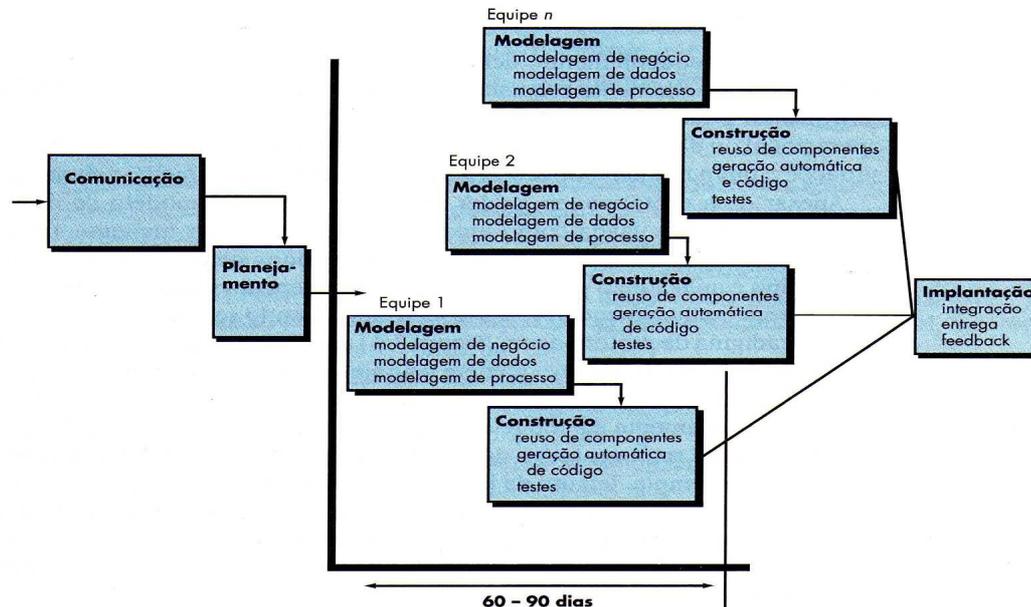
Figura 5. Modelo RAD

[Fonte: retirado de [4]]

As desvantagens desse modelo são:

- Necessidade de alto número de recursos humanos para desenvolver as funções em tempo hábil;

- Comprometimento dos desenvolvedores e clientes;

- Necessidade de alta modularização do sistema, o que consome muito tempo de reunião, diminuindo o tempo de desenvolvimento;

- Devido ao estilo de alta velocidade de desenvolvimento, pode não ser adequado quando existem altos riscos.

2.3 Modelos evolucionários de software

Com o passar do tempo, o software evolui como todo sistema complexo. Os requisitos mudam, os prazos para entregar o produto final ficam cada vez mais curtos. Com isso, os processos de software se adaptaram de modo que se entregue uma versão reduzida, mas que atenda todos os requisitos básicos, num prazo curto que permita seu uso pelos clientes.

A ideia por trás do desenvolvimento evolucionário é implementar um sistema básico, uma versão inicial, mas que, ao passar do tempo, outras versões, com funcionalidades mais complexas, sejam entregues para completar o projeto.

A abordagem evolucionária do desenvolvimento de software, muitas vezes, é mais eficaz do que a abordagem cascata, no sentido de produzir sistemas que atendam às necessidades imediatas dos clientes [7].

O modelo evolucionário tem como vantagem o fato da especificação do sistema ser feita gradativamente. À medida que o usuário adquire maior conhecimento em relação às suas preferências no sistema, compreendendo melhor seus problemas, isso passa a ser refletido no software.

Serão brevemente apresentados dois processos evolucionários: prototipagem, modelo espiral.

2.3.1 Prototipagem

Muitas vezes o cliente não tem uma clara visão dos requisitos que lhe interessam, ou o desenvolvedor não tem certeza sobre o uso de um certo algoritmo ou sobre a portabilidade para um sistema operacional, ou qual interação homem/máquina o sistema deve assumir. Nesses casos, o uso da prototipagem se faz muito útil.

Primeiramente, o engenheiro de software se reúne com o cliente a fim de delinear os requisitos básicos do sistema, as necessidades conhecidas e requisitos que precisam ser mais bem definidos. A seguir, o projeto se concentra em entregar um executável com todos os requisitos básicos e uma interface próxima do ideal. O

feedback do cliente vai servir para aprimorar esses requisitos e implementar outros mais complexos, tornando o entendimento por parte do desenvolvedor mais fácil. A Figura 6 ilustra como o processo de prototipagem acontece dentro do projeto.

Idealmente, o protótipo serve apenas para identificar os requisitos que estão sendo satisfeitos e como o sistema pode ser aprimorado. Mas muitos clientes usam esse protótipo inicial como se fosse o executável final, ao perceberem que não é o produto final, eles reclamam e exigem consertos, ao invés de esperarem por uma nova versão mais refinada do sistema, com mais qualidade e robustez.
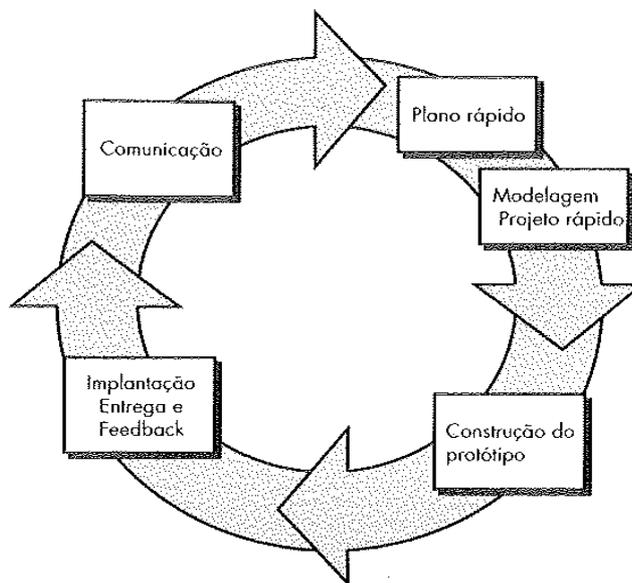
Figura 6. Ciclo da Prototipagem

[Fonte: retirado de [4]]

Apesar desses problemas, o paradigma de prototipagem pode ser muito efetivo para os engenheiros de software. O importante é definir as regras de desenvolvimento logo no início do projeto (linguagem e algoritmos) assim como entender todos os requisitos do cliente.

2.3.2 Modelo espiral

O modelo espiral foi proposto por Boehm, em 1988 [4]. É um modelo de processo de software que combina a natureza iterativa da prototipagem e os aspectos controlados do desenvolvimento em cascata. Ao invés de representar o

processo como uma sequência de atividades, cujo retorno é essencial para iniciar a próxima atividade, o modelo é representado como uma espiral.

Cada loop da espiral é divido em quatro setores:

- **Definição de objetivos** – São definidos os objetivos específicos do projeto. Também são identificadas as restrições para o processo e o produto e é preparado um plano de gerenciamento. Os riscos também são identificados e, dependendo desses riscos, estratégias alternativas são planejadas.

- **Avaliação e redução de riscos** – Após os riscos serem identificados, uma análise detalhada dos mesmos é feita e providências são tomadas para reduzi-los.

- **Desenvolvimento e validação** – Após a avaliação dos riscos, um modelo de desenvolvimento é escolhido para o projeto. Por exemplo, se o risco principal identificado na fase anterior for a integração de sistemas, o modelo cascata pode ser adotado.

- **Planejamento** – O projeto é revisado para que seja tomada uma decisão a respeito do próximo loop da espiral. Caso seja decidido continuar, novos planos serão traçados para o projeto seguindo a mesma metodologia.

A Figura 7 ilustra como ocorre um desenvolvimento de projeto seguindo o padrão espiral de projeto. Observar que a cada nova espiral realizada, novos planejamentos são feitos para o projeto.
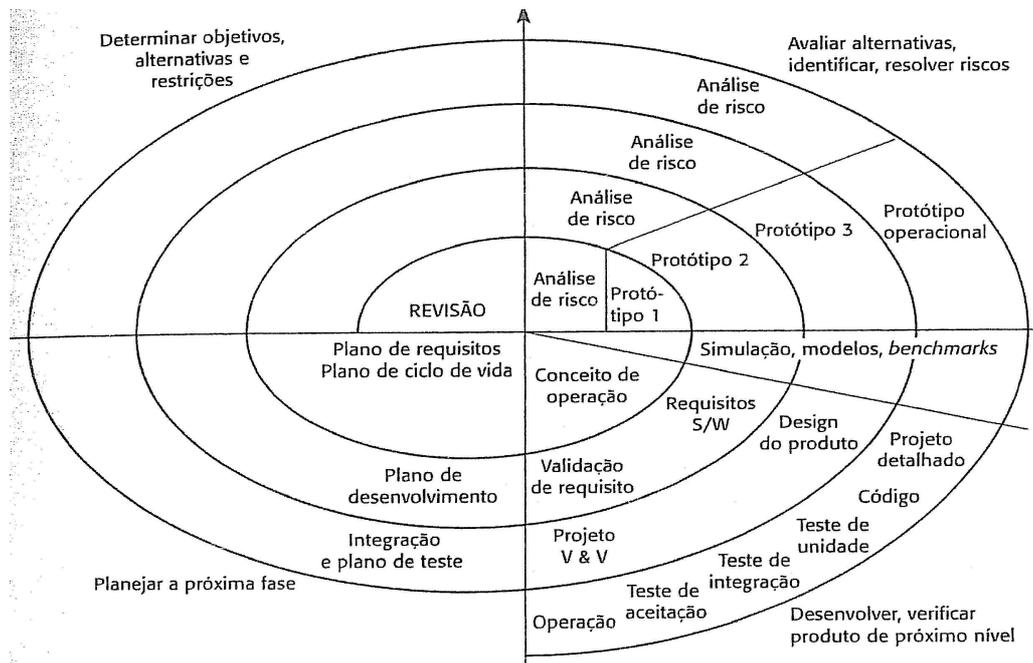
Figura 7. O modelo espiral

[Fonte: retirado de [7]]

Uma importante diferença do modelo espiral para ou outros modelos de processo, exceto o RUP, é a explícita consideração dos riscos no modelo espiral. Os riscos podem resultar em problemas no projeto, como ultrapassar o prazo ou o custo previsto.

Diferentemente dos outros processos, que são encerrados quando o software é entregue, no modelo em espiral existe uma constante evolução do produto, tornando o mesmo mais robusto e mais de acordo com os pedidos do cliente.

O modelo espiral é uma abordagem realista do desenvolvimento de software e sistemas de grande porte [4]. Como o software evolui, tanto o desenvolvedor como o cliente podem entender melhor o sistema e terem a reação adequada aos riscos de cada nível evolucionário. O modelo usa a prototipagem como uma ferramenta de redução de riscos, como exibido na Figura 7, porém, pode permitir que a prototipagem seja usada em mais de um estágio da evolução do produto.

Para convencer os clientes de que a abordagem espiral é controlável, deve-se ter uma competência considerável para avaliar os riscos. Caso um risco grave não seja identificado no início, certamente ocorrerão problemas.

2.4 Desenvolvimento ágil

A partir da década de 90, começaram a surgir novos métodos sugerindo uma abordagem de desenvolvimento ágil onde os processos adotados tentam se adaptar às mudanças, apoiando a equipe de desenvolvimento no seu trabalho.

Os métodos ágeis caracterizam-se pelo seu caráter adaptativo e orientado a pessoas. As abordagens ágeis compartilham, na sua essência, o processo de desenvolvimento centrado nas pessoas, orientado à obtenção de artefatos a partir de iterações, o que, consequentemente, impõe o caráter adaptativo durante todo o ciclo de desenvolvimento.

No início da primeira década de 2000, um grupo de especialistas formulou um conjunto de princípios que definem a metodologia ágil [1]. Estes são os seguintes:

- A prioridade é satisfazer o cliente através de entregas contínuas e frequentes;
- Receber bem as mudanças de requisitos, mesmo em uma fase avançada do projeto;
- Entregas com frequência, sempre na menor escala de tempo;
- As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente;
- Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários;
- A maneira mais eficiente da informação circular dentro do time de desenvolvimento é através de uma conversa presencial;

- Ter o sistema funcionando é a melhor medida de progresso;

- Processos ágeis promovem o desenvolvimento sustentável;

- Atenção contínua à excelência técnica e a um bom design, aumenta a agilidade;

- Simplicidade é essencial;

- As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;

- Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz.

Com base nesses princípios, as metodologias ágeis existentes apresentam um conjunto de atividades a serem adotadas durante o desenvolvimento de sistemas. O termo “metodologias ágeis” tornou-se popular em 2001 quando dezessete especialistas em desenvolvimento de software representando os métodos Scrum, *Extreme Programming* e outros [6], estabeleceram os princípios comuns, anteriormente listados, compartilhados por esses métodos.

Os processos ágeis mais conhecidos são os seguintes:

- *Extreme programming* (XP): É uma abordagem deliberada e disciplinada para o desenvolvimento do software. O primeiro projeto a usar o XP foi o C3, da Chrysler [4]. É uma metodologia baseada em quatro categorias de regras e práticas a fim de garantir a qualidade do software produzido:

  - **Planejamento** – O cliente descreve suas atividades que serão absorvidas pelo software, conhecidas como estórias do usuário. O cliente, então atribui um valor a essas estórias, que definem as prioridades das funcionalidades do sistema. A equipe XP, então, atribui um custo a cada estória, medido em semanas de desenvolvimento. Caso o custo de uma estória ultrapasse três semanas, pede-se que o cliente divida a estória original em estórias menores e o processo de atribuição de custos e valor acontece novamente. Uma observação importante, novas estórias podem ser escritas a

qualquer momento do processo. Uma vez feito um compromisso entre equipe e cliente de quais estórias irão entrar na versão do software e a data de entrega dessa versão, a equipe estuda qual o modo seguir:

- Todas as estórias serão implementadas;
- As estórias de maior prioridade serão implementadas primeiro;
- As estórias com maior risco serão implementadas primeiro.

Após a entrega da primeira versão, a equipe calcula a velocidade do projeto de acordo com a quantidade de estórias que foram entregues na primeira versão. Essa medição ajuda em dois pontos: em relação ao tempo de entrega do sistema; e em relação ao comprometimento (esforço) da equipe dentro de uma versão. Se ocorrer um comprometimento excessivo dos profissionais, haverá uma modificação no conteúdo das versões ou na data de entrega.

- **Projeto** – Segue rigorosamente o princípio KIS (*keep it simple* – mantenha a simplicidade). Fornece diretrizes de implementação para uma estória a partir da maneira como ela foi escrita. Se for encontrado um problema difícil nessa fase, é recomendada a construção de um protótipo operacional, que não será uma versão final do sistema. Esse protótipo é denominado solução de ponta, e será avaliado a fim de diminuir o risco de uma estória quando a verdadeira implementação for iniciada e revisar as estimativas de entrega das versões.

- **Codificação** – Tem como característica chave a programação em pares, às vezes até uma terceira pessoa entra no processo de codificação. Essa característica é importante, pois mantém os desenvolvedores concentrados no problema a ser resolvido. Na prática cada um pode assumir uma tarefa diferente durante a codificação. Por exemplo, um desenvolvedor fica responsável pelos cálculos a serem escritos no código enquanto outro desenvolvedor fica responsável pela adequação do código aos critérios de codificação, garantindo que esse código encaixe no projeto mais amplo da estória. Isso faz com que a funcionalidade de uma estória seja entregue no

tempo correto, seguindo todos os padrões do projeto e com qualidade para o cliente.

➢ **Testes** – É recomendado que, ao terminar de projetar as estórias do cliente, sejam criados testes unitários, os quais exercitarão a interface, as estruturas de dados e a funcionalidade de um componente do projeto. Essa prática encoraja a prática de testes de regressão: testar as funcionalidades já implementadas, sempre quando o código for modificado. À medida que os testes unitários são realizados e organizados numa sequência, o teste de integração pode ser feito diariamente. Isso fornece à equipe uma visão de progresso e indica sinais de alerta caso o projeto tenha problemas. Ao final são feitos os testes de aceitação, estes feitos pelo cliente, que são testes das estórias do usuário implementadas numa versão do sistema.

• SCRUM: Desenvolvido na década de 1990 por Jeff Sutherland e sua equipe. O nome deriva de uma atividade decorrente do jogo de *rugby*. Esse método orienta-se por três princípios: a visibilidade, a inspeção e a adaptabilidade. Todos os requisitos devem estar visíveis a todos os envolvidos no desenvolvimento, a inspeção deve ser uma ação frequente e, consequentemente, as ações para adaptação do produto de software devem ser realizadas. No SCRUM, são incorporadas as seguintes atividades de desenvolvimento: requisitos, análise, projeto, evolução e entrega. Essas atividades são reunidas em um padrão de processo, chamado *Sprint.*

Dentro de uma *Sprint*, o problema do cliente é adaptado, definido e executado, podendo sofrer modificações durante a *Sprint*. O tempo da *Sprint* é definido pelo responsável do projeto, podendo ter 30 dias ou uma semana, na Figura 8, este tempo é definido em 30 dias.

A cada dia da *Sprint*, é feita uma reunião rápida de 15 minutos para discutir brevemente o que foi feito no dia anterior, o que está atrapalhando o desenvolvimento da nova funcionalidade e o que será feito até a próxima reunião rápida (*stand up meeting*), que acontece no dia seguinte, preferivelmente sempre no mesmo horário.

No final da *Sprint,* uma versão de demonstração com as funções definidas no início é apresentada ao cliente e o mesmo realiza a avaliação. Com o SCRUM, uma equipe de desenvolvimento pode trabalhar de modo que os problemas possam ser previamente vistos, ocasionando novas reuniões de projeto para sanar esses problemas, tornando o desenvolvimento mais bem sucedido.
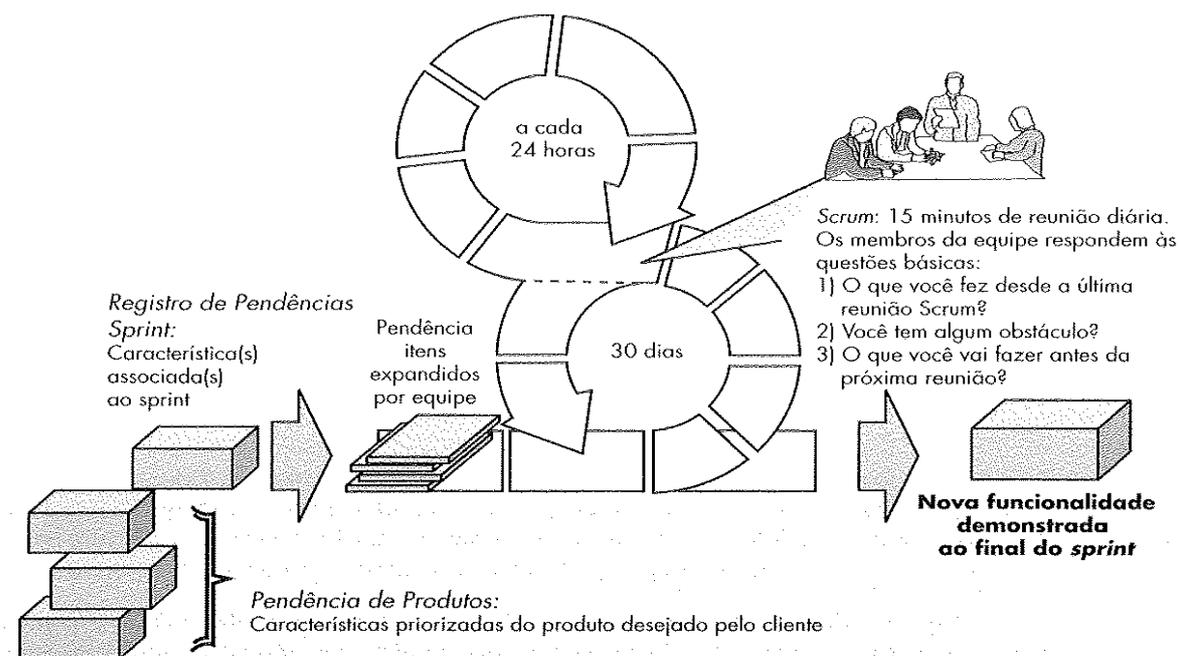
Figura 8. Metodologia Scrum

[Fonte: retirado de [4]]

- FDD (*Feature Driven Development* – Desenvolvimento Guiado por Características): Foi concebido como um modelo prático de processo de engenharia de software orientada a objetos. Em seu contexto, uma característica seria uma função valorizada pelo cliente que poderia ser implementada em duas semanas ou menos. Esse método se baseia nos seguintes processos: desenvolver um modelo global, construir uma lista de características, planejar por características, projetar e construir por características. Estes processos são ilustrados na Figura 9. O FDD dá mais ênfase em diretrizes e técnicas de gestão de projeto do que os outros métodos ágeis [4].
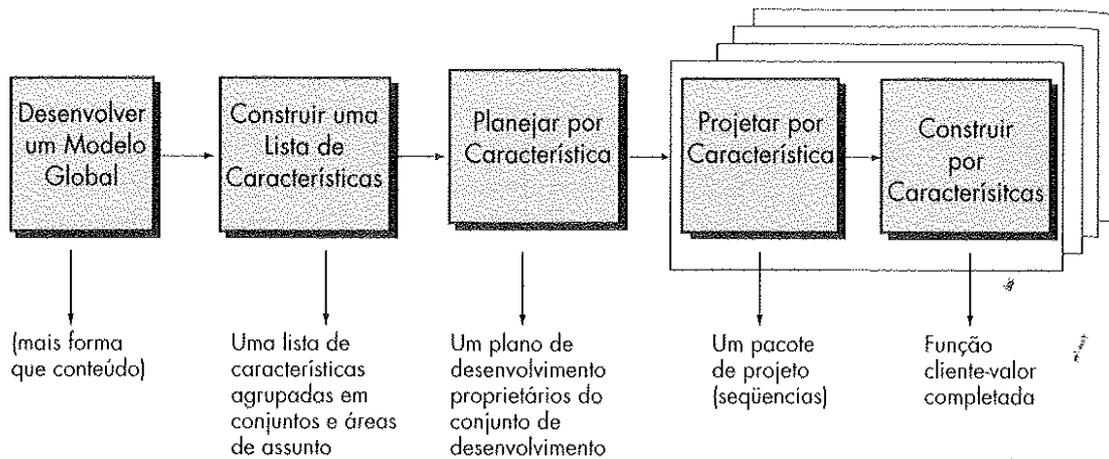
Figura 9. Desenvolvimento guiado por características

[Fonte: retirado de [4]]

Desta forma, conclui-se a apresentação dos principais modelos de desenvolvimento de software. No próximo capítulo, estes modelos serão comparados entre si, destacando os pontos positivos e negativos de cada um deles. Ou seja, em quais situações devemos preferir um ou outro modelo.

Capítulo 3
Comparação entre metodologias ágeis e metodologias clássicas

3.1 Introdução

Nesse capítulo serão detalhadas vantagens das metodologias ágeis em relação às metodologias clássicas de engenharia de software.

Com essa comparação, é mostrado o porquê da maior adoção das metodologias ágeis nas empresas especializadas em desenvolvimento de software a fim de acelerarem a área de manutenção de sistemas. Com isso, as empresas aumentam a satisfação dos clientes, podendo alcançar maiores ganhos financeiros e de novos clientes.

3.2 Comparativo entre as metodologias

A maioria das metodologias ágeis nada possui de novo [3]. A primeira diferença das metodologias ágeis para as tradicionais é a priorização das pessoas e não dos documentos. Com isso, as metodologias ágeis conseguem ser mais dinâmicas às mudanças de requisitos, pelo fato de realizarem reuniões rápidas para definir as soluções a serem implementadas.

Outras diferenças aparecem no que diz respeito ao início do projeto, planejamento, execução, controle e encerramento do mesmo. Essas diferenças serão apontadas ao longo do capítulo, ilustrando as vantagens e desvantagens entre as metodologias.

3.2.1 Início do projeto

Na metodologia tradicional, os objetivos e o escopo devem ser previamente estabelecidos [5]. Após identificar todos os elementos do sistema, esse escopo inicial do projeto deverá descrever todas as principais funções que o sistema deverá realizar. Neste momento, também serão identificadas soluções alternativas e restrições técnicas e administrativas do projeto.

A dificuldade encontrada nesse modo de iniciação de projeto se dá ao fato que não se pode ter certeza em relação ao custo e ao prazo de conclusão do projeto. Prever o tempo que será gasto para desenvolver as funções sem possuir informações detalhadas a respeito das atividades torna a atividade complicada. A ideia de especificar totalmente um software antes do início de sua implementação é extremamente difícil [2].

Nas metodologias ágeis, a iniciação do projeto começa pela identificação dos objetivos do cliente, mapeando suas estórias em elementos do produto [5]. A obtenção dos requisitos se dá junto ao cliente, escutando suas estórias, a fim de criar seus requisitos de maneira superficial, porém objetivos, atendendo as expectativas do cliente.

Esses requisitos são definidos sem muitos detalhes, já que, na metodologia ágil, eles podem ser alterados dinamicamente pelo cliente. Outra preocupação é em relação aos requisitos se tornarem obsoletos à realidade do cliente. A equipe de desenvolvimento precisa estar preparada para tais mudanças. Essa falta de preparo é uma das principais causas de falha nos projetos de softwares, portanto uma equipe preparada para as mudanças se torna mais eficiente e menos sujeita a fracassos.

Na Figura 10, são destacados os pontos principais de início de projeto que são considerados pelas duas metodologias. Enquanto a metodologia tradicional prioriza o escopo, o prazo e o custo, sacrificando a qualidade do sistema, o que pode deixar o cliente insatisfeito com a empresa; a metodologia ágil tem foco na qualidade, no prazo e no custo. O destaque está na qualidade, que, na metodologia

ágil, não é negociável. O produto sempre deve ter uma boa qualidade para garantir a satisfação do cliente.
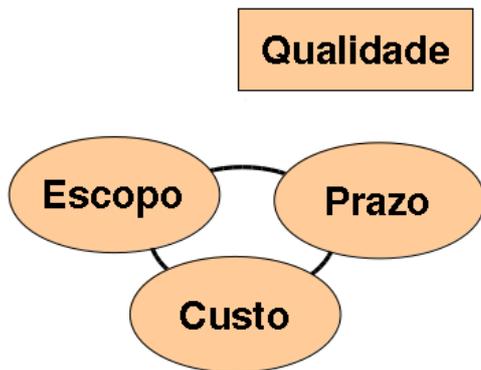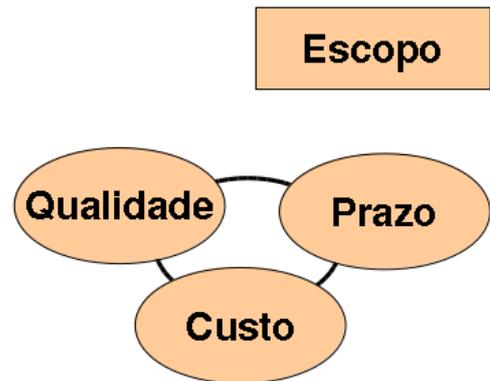
Figura 10. Fatores de projeto

[Fonte: retirado de [5]]

3.2.2 Planejamento

Na fase de planejamento, muitos fatores são considerados:

- Escopo;
- Qualidade;
- Tempo;
- Riscos;
- Recursos humanos;
- Custos.

Serão detalhadas as comparações entre as metodologias ágeis e tradicionais para cada um desses fatores apresentados, esclarecendo as vantagens e desvantagens entre as mesmas.

3.2.2.1 Escopo

No modelo tradicional, a aquisição dos requisitos é intensa nessa fase. A equipe de desenvolvimento fará entrevistas com o cliente e o usuário final, obtendo todas as informações a respeito dos processos de negócio, as funções que o software irá realizar. Nessa fase não pode haver erros, como interpretações errôneas e informações falsas. Isso pode ocasionar o fracasso do projeto.

Os resultados obtidos através dessas entrevistas, e após análise, deverão ser documentados para que outros analistas - de prazo, custos e riscos – possam usá-los para a posterior análise.

Na metodologia ágil, ocorre um planejamento superficial dos requisitos, no início do projeto, com a finalidade de informar a equipe de análise de requisitos todas as estórias identificadas.

Diferentemente da metodologia tradicional, a metodologia ágil detalha o escopo a cada iteração – no caso do Scrum, a cada *Sprint*. Essa forma de detalhamento é feita sem a obrigatoriedade de documentação, de forma interativa, junto com o cliente.

Ao final, percebe-se que o planejamento sai da fase inicial do processo e passa a ser executado a cada iteração, na fase de execução, ao longo do projeto.

3.2.2.2 Tempo

Na metodologia tradicional, o planejamento do tempo é baseado na separação das atividades do projeto, desde o início até a entrega ao cliente, organizando-as numa sequência coerente e realizando a estimativa de duração.

Ainda assim, é uma atividade muito complicada, pois não é possível prever precisamente todos os detalhes e quais atividades estarão sendo feitas no futuro. Essas estimativas precisam ser revistas ao longo do projeto, caso apareçam novas informações durante o desenvolvimento.

Um detalhe importante no desenvolvimento de sistemas é que o principal fator de influência é o ser humano. Então a estimativa de prazo precisa respeitar os

aspectos humanos. Por exemplo, caso algum desenvolvedor fique doente, tenha problemas particulares graves, entre outras possibilidades. Por esses motivos, geralmente o gerente de projeto cria estimativas com prazos um pouco maiores para ter algum tempo de folga, caso apareçam problemas de fator humano no projeto. Mas deve-se tomar cuidado para que essas estimativas não ultrapassem o prazo de entrega do produto, muitas vezes acordado no início do projeto.

Já na metodologia ágil, ocorre uma divisão de atividades de acordo com a importância das mesmas em relação ao cliente. As mais importantes são escolhidas para serem desenvolvidas primeiro então esses requisitos são divididos em pequenas tarefas que serão desenvolvidas dentro de uma iteração.

O sequenciamento de atividades não existe na metodologia ágil [5]. Nessa metodologia, diferentemente da metodologia tradicional, não é feito um planejamento de todas as atividades que serão executadas no projeto. O planejamento é feito de forma dinâmica ao longo da iteração.

Em casos onde as empresas que adotam a metodologia ágil para a parte de desenvolvimento, mas que se envolvem em negócios onde é preciso exibir os custos antecipadamente, por exemplo, numa licitação, ocorre um uso de metodologias híbridas. Nesses casos, é comum o uso de metodologia tradicional no que se diz respeito à documentação do processo e à base de custos. Nessas metodologias híbridas, o processo de documentação é importante para passar ao cliente as estimativas de preço e custo, a fim de que o mesmo aprove ou realize eventuais revisões.

3.2.2.3 Qualidade

Na metodologia tradicional, o plano de qualidade é feito pela equipe de gerência de qualidade do projeto. Nesse plano estão listados os atributos de qualidade do projeto. Entre esses atributos estão: segurança, confiabilidade, modularidade, portabilidade, facilidade de uso e outros. Dependendo da finalidade do software, o seu desenvolvimento será executado de acordo com a ordem de importância dos atributos.

É necessário definir a importância de cada atributo, evitando que a equipe de desenvolvimento utilize força mais que necessária para executar o projeto. Entende-se força mais que necessária como tecnologias que não precisam estar no projeto.

Também é considerado um atributo de qualidade de software a conformidade das funções do mesmo com as necessidades do cliente. Mesmo que o software preencha vários atributos como segurança, confiabilidade e modularidade, um software que não atende as necessidades do cliente é considerado de má qualidade.

Na metodologia ágil, o planejamento de qualidade, mais uma vez, deixa de ser feito no início do projeto, o que levaria à previsão de ações num futuro distante e passa a ser feito iterativamente, durante o desenvolvimento, visando à aproximação com as necessidades do cliente.

Nessa metodologia, cada requisito, e não atributos de qualidade, é avaliado quanto à sua importância. É a chamada avaliação externa do produto, feita pelo cliente, que ter por finalidade obter do cliente as características que mais lhe interessam.

Existe também a qualidade interna do software. Esta análise é feita pela equipe de desenvolvimento do software. Nesse tipo de qualidade, são vistos os aspectos de como o software deve ser desenvolvido, quais técnicas e qual o prazo para a entrega do mesmo.

A qualidade interna deve ser discutida entre o cliente, geralmente os responsáveis pela área de T.I. do cliente, e os desenvolvedores durante o planejamento de cada iteração. O cliente pode interferir no aspecto de segurança e portabilidade, a fim de atender às suas necessidades, mas quanto ao prazo, isso fica exclusivamente sob responsabilidade dos desenvolvedores. O prazo é dito com atributo não negociável.

Nesse ponto, destaca-se a observação de que as metodologias tradicionais usam o controle para medir a qualidade do desenvolvimento. Já as metodologias ágeis usam uma abordagem com foco na qualidade durante o desenvolvimento.

3.2.2.4 Riscos

Os riscos, dentro da metodologia tradicional, são identificados e as suas soluções será incorporadas ao plano de projeto. Em muitos casos, uma equipe de desenvolvimento tem capacidade de identificar os riscos durante o período inicial do desenvolvimento, mas prever riscos que poderão acontecer ao longo dos anos é uma tarefa extremamente complicada.

Na metodologia ágil, para diminuir riscos de desenvolvimento, o cliente começa a fazer parte da equipe de trabalho. Ele ajudará nos problemas relacionados ao escopo do projeto, pois, com suas estórias, o especialista pode identificar um risco grande, trazendo sua solução já para o início do projeto.

Nas metodologias ágeis, não se faz uso de documentos contendo um plano formal de gerência de riscos. Esses riscos são abordados através das reuniões diárias realizadas pela equipe de desenvolvimento.

3.2.2.5 Recursos Humanos

As pessoas que irão trabalhar na equipe de desenvolvimento do software serão contratadas pela equipe de gerência de recursos humanos. É desejável que sejam contratadas pessoas com grande experiência e possuam conhecimentos apropriados para o projeto. Porém, existe a possibilidade do orçamento ser limitado, fazendo com que o gerente de recursos humanos contrate pessoas sem muita experiência, mas também com pouca remuneração.

O problema de contratar pessoas sem muita experiência consiste no aparecimento de pequenos erros ao longo do projeto que podem aumentar os riscos relacionados ao tempo, qualidade e custo do projeto.

Grupos de projetos de software não devem ter mais que oito participantes, devido à dificuldade de grandes equipes trabalharem de maneira eficaz para resolver um único problema [7]. Com isso, dividindo a equipe, os problemas de comunicação são diminuídos.

Na metodologia tradicional, a equipe tem uma divisão bem clara de funções. Analista de sistemas, programador e testador são os cargos de uma equipe nessa metodologia. Cada um participa exclusivamente da fase de projeto que lhe diz respeito. O uso de pessoas especializadas em uma única fase do projeto facilita o caso de substituição, utilizando a mão de obra disponível do mercado.

Na metodologia ágil, o desenvolvedor exerce todas as funções listadas anteriormente. Com isso, ele pode ser usado em diversas etapas do projeto. O problema é que com essa acumulação de funções, os profissionais dessa metodologia precisam ser de alta capacidade, capazes de definir a melhor forma de desenvolvimento e testes.

O desenvolvedor, nas metodologias ágeis é multi-capacitado, sendo de difícil substituição. Por outro lado, essas metodologias incentivam o aprimoramento do profissional, tornando-o mais importante dentro de uma equipe.

3.2.2.6 Custos

Na metodologia tradicional, os custos já são alocados no início do projeto, a partir das tarefas designadas às equipes de desenvolvimento e análises de estimativas. Portanto é elaborada uma base de custos que será acompanhada durante a execução do projeto com a finalidade de comparar os custos atuais com os custos iniciais.

Na metodologia ágil, o custo do projeto é fixo e só irá variar caso o cliente solicite uma renovação do contrato. Essa renovação está associada ao fato do cliente estar satisfeito com a empresa ou com alguma alteração no escopo de projeto corrente solicitada pelo cliente.

A alocação de recursos não é feita no início do projeto e sim, durante a realização do mesmo, a cada iteração. O desenvolvedor tem a livre escolha para executar as tarefas do projeto em função do andamento do desenvolvimento. Como ele tem uma visão mais técnica, é o melhor a tomar essa decisão.

3.2.3 Execução

Nas metodologias tradicionais, ao iniciar a execução de um projeto, deve-se seguir exatamente o que está definido no plano de projeto, definido na fase de planejamento. A equipe de desenvolvedores segue essa documentação e suas equipes são supervisionadas pela equipe de gestão de qualidade de modo que os padrões definidos sejam respeitados.

As equipes são divididas em funções específicas, como analistas, programadores e testadores. Devido a essa divisão, uma capacitação pode ser usada para desenvolver uma equipe que está apresentando um baixo índice de produtividade, para não comprometer o prazo de entrega do projeto. Existe também um problema com a interação entre os membros da equipe, dificultando a disseminação do conhecimento sobre o projeto e sobre as tecnologias usadas durante a execução do mesmo.

Nas metodologias ágeis, a execução do projeto ocorre de forma mais dinâmica, onde os requisitos que serão desenvolvidos são discutidos em reunião na fase inicial de cada iteração. Ocorre uma maior integração entre os membros da equipe, facilitando a comunicação e, com isso, aumentando o conhecimento a respeito do projeto, tornando o profissional mais qualificado para projetos posteriores.

3.2.4 Controle

Nas metodologias tradicionais, existe um rígido controle quanto às mudanças no escopo do projeto. Qualquer alteração reflete como um risco a todo o projeto, exigindo uma análise de impacto em todas as áreas que serão afetadas. Quanto ao controle do tempo, existe um cronograma a ser seguido e tradicionalmente é usado o gráfico de Gantt como referência para saber se o desenvolvimento do projeto está obedecendo ao tempo previsto no planejamento inicial.

Os custos do projeto são monitorados através da equipe de gerência de custos, a qual faz uma comparação com a linha de base de custos do projeto. Uma não conformidade dos custos pode trazer a falta dos mesmos para a execução do resto do projeto, implicando até em sua paralisação para novas negociações.

Por fim, a qualidade é controlada com o uso de documentação periódica, escrita pela equipe de gerência de qualidade. Caso existam não conformidades, a equipe gera um relatório que será enviado às áreas responsáveis a fim de sanar os problemas.

Na metodologia ágil, não existe um controle rígido sobre o escopo. As mudanças ficam a cargo do cliente. Essas mudanças são vistas como algo positivo, pois trazem mais aprendizado à equipe de desenvolvimento. No Scrum, por exemplo, o controle do tempo se dá com o auxílio de um gráfico, chamado *Burndown*. Esse gráfico apresenta quais requisitos foram concluídos e se o tempo de execução está obedecendo ao estipulado na reunião inicial da iteração. Caso um requisito leve mais tempo que o necessário, os outros membros da equipe realizam um maior esforço, de modo que esse requisito seja finalizado dentro dessa interação.

Nessa metodologia, não existe um controle rígido dos custos, pois com as mudanças de escopo definidas pelo cliente, esses custos podem tanto aumentar como diminuir. Ficará a cargo do cliente a gerência dos custos, pois este está participando ativamente da equipe de desenvolvimento.

Em alguns casos, os clientes não querem pagar mais do que já foi estipulado no início do projeto, como nos casos de uso híbrido de metodologias. Nessas circunstâncias, a estimativa de tempo não pode ter muitos erros, para não ocasionar altos custos ao cliente. Existem ainda casos, como quando o sistema precisa de uma correção, onde a empresa assume os custos para que o cliente não se sinta insatisfeito com a empresa.

3.2.5 Encerramento

Nas metodologias tradicionais, no momento de encerramento de um projeto, os participantes serão informados através de um documento formal. Serão mostrados os indicadores de desempenho e de qualidade medidos durante a execução do projeto, e essas informações serão levadas aos responsáveis pelo patrocínio do mesmo.

Já na metodologia ágil, será feita uma reunião, onde serão discutidos os erros e acertos ocorridos durante o projeto. Esta é definida com reunião de retrospectiva, no caso do Scrum. Essa reunião tem como objetivo fortalecer o aprendizado entre os membros da equipe. Também é importante a participação do cliente para elevar a moral da equipe, comemorando a entrega do projeto. Normalmente, não são gerados documentos de entrega ao final do projeto.

Capítulo 4
Estudo de caso – Sistema de Gestão Empresarial Pirâmide

Nesse capítulo, será demonstrado como o uso de metodologias ágeis ajudou o desenvolvimento em uma *software house* de modo que a alta demanda dos clientes fosse suprida. Essas metodologias também ajudaram na gerência de uma maior quantidade de pessoas que foram contratadas para as equipes de desenvolvimento.

A seguir será feita uma apresentação da empresa Procenge e o sistema estudado, o Pirâmide. Após essa apresentação, serão sintetizadas as opiniões dos especialistas da área de qualidade de software, como também a opinião de profissionais que antes trabalhavam usando somente a metodologia tradicional na área de manutenção e passaram a utilizar a metodologia ágil, especificamente o Scrum, para terem maior rendimento na produtividade.

4.1 A Procenge

A Procenge surgiu há 40 anos com o propósito de desenvolver sistemas de gestão administrativa e de apoio à tomada de decisões. Está instalada no Porto Digital do Recife desde 1998. Nesses anos de sua história, desenvolveu parcerias com empresas privadas e públicas. Atualmente o maior foco da Procenge está nos setores de saneamento, sucroenergético (usinas de cana-de-açúcar), saúde e gás.

Desde 2002, a Procenge passou a adotar normas exigidas pelas certificações de qualidade. Com isso, implantou o SGP – Sistema de Gestão Procenge, que redefiniu as orientações estratégicas com definições de processos, práticas e aplicação de recursos.

Após essa reestruturação na sua gestão, a Procenge foi certificada com os selos de qualidade ISO 9001:2008, CMMI nível 2 e MPS.Br nível F. O sistema de gestão passa por revisões periódicas, adaptando-se às mudanças exigidas por essas instituições de qualidade.

4.2 O Pirâmide

O sistema Pirâmide é o principal produto da Procenge. Esse sistema proporciona que os diretores e gerentes obtenham todas as informações necessárias ao funcionamento da empresa de modo interligado. Oferecendo, assim, agilidade e transparência nos processos de gestão administrativa, do financeiro ao setor de compras, do fiscal ao estoque. Por ser constituído de vários módulos, a manutenção do sistema se tornou muito difícil de ser gerido por apenas um gerente de projeto.

O Pirâmide é uma solução ERP (*Enterprise Resource Planning*) que se aplica a empresas de todos os setores da economia. Ele incorpora aos seus módulos aspectos fundamentais da gestão empresarial, valorizando a integração de dados e flexibilidade de adaptação a empresas dos mais variados segmentos. O software automatiza e integra os processos de diversas áreas da organização, tais como: contabilidade, controladoria, financeira, suprimentos, produção, comercial, logística e RH, facilitando o fluxo de informações entre elas.

4.3 Estudo de caso

O caso estudado foi como a incorporação da metodologia ágil no processo de manutenção e evolução do Pirâmide ajudou as equipes de desenvolvimento a terem maior produtividade e aquisição de conhecimento técnico e de negócios, tornando-se mais especializados.

Anteriormente, usando somente a metodologia tradicional, especificamente a metodologia em cascata, toda a área de manutenção era gerida a partir de documentos de especificação e implementação. A fábrica do Pirâmide conseguia

suprir a demanda dos clientes no que diz respeito às correções e novas funcionalidades, mas o tempo de desenvolvimento estava se tornando alto, justamente pelo alto uso de documentação de projeto, principalmente na especificação dos requisitos e na implementação.

A partir de então, começou a ser adotado um sistema de gestão, onde todos os procedimentos de atendimento do Pirâmide foram mapeados, documentados e organizados. A esse sistema foi dado o nome de SGP – Sistema de Gestão Procenge. A Figura 11 ilustra de modo geral como é composto o sistema.

Figura 11. Sistema de Gestão procenge

[Fonte: Intranet Procenge]

O SGP contém todos os procedimentos os quais os profissionais contratados pela Procenge devem tomar como referência para o desenvolvimento do Pirâmide. O sistema está estruturado conforme as normas da ISO 9001. No topo dessa estrutura de documentação está o manual da qualidade, que é um documento estratégico da organização, e que apresenta o modelo de gestão da empresa. Neste estão contemplados a política e os objetivos da qualidade, as autoridades e responsabilidades, o comprometimento da alta direção, o escopo, os indicadores de acompanhamento de objetivos e processos e o mapeamento dos processos do SGP, assim como referências aos procedimentos que orientam a operacionalização desses processos.

Após a criação desse sistema, a gerência de qualidade tomou a decisão de incorporar a metodologia ágil, especificamente o Scrum, no processo de manutenção e evolução do Pirâmide.

A partir do uso de metodologia ágil, o tempo entre receber a solicitação de mudança do cliente e a entrega da versão final diminuiu. Em termos de medição, uma mudança que levaria duas semanas para ser entregue usando a metodologia em cascata, está sendo entregue em uma semana; pois, com as reuniões de planejamento feitas no começo de cada iteração (*Sprint*) o entendimento fica mais disseminado entre os membros da equipe.

O software também se tornou mais robusto e confiável com o uso do Scrum, pois todos os membros da equipe adquirem maior conhecimento para realizar os testes necessários à identificação de *bugs*.

Outra vantagem trazida pelo Scrum à fábrica foi a descentralização da gerência de projetos. O acompanhamento começou a ser feito pelos *Scrum Masters*; assim, o gerente de projetos deixou de acompanhar cada desenvolvedor individualmente. Ele agora acompanha as equipes, que passaram a distribuir o escopo de maneira interna, tornando-as auto gerenciáveis, o que aumentou a responsabilidade dos membros.

Por último, houve também a disseminação do conhecimento de negócio do Pirâmide entre os membros da equipe. Hoje não é somente a pessoa que irá desenvolver uma solicitação que deve entender o que foi pedido, mas todos na equipe de desenvolvimento devem entender, caso seja necessário ajudar na entrega desse pedido em um tempo mais curto do que foi inicialmente estimado.

O que ocorre nesse projeto é, na verdade, uma 'hibridização' de metodologias. O processo se inicia com uma solicitação do cliente à empresa referente a uma modificação que se queira realizar no software. Então, esse pedido é encaminhado ao especialista de produto, que é um profissional com alto conhecimento de práticas de mercado, para que ele encontre uma solução que obedeça aos padrões do sistema.

Após essa solução ter sido documentada num documento de requisitos tradicional, a mesma é encaminhada ao analista de sistemas responsável pela área do sistema em que essa solução fará parte. Este então faz a estimativa de tempo e esforço, a qual indicará os valores a serem cobrados pela Procenge ao cliente. Somente após a validação dessa documentação de estimativa e custos pelo cliente, é que a metodologia ágil começa a ser usada.

Com o Scrum, essas requisições de cliente validadas ficam dentro do *backlog* de produto, no qual são organizados em ordem de prioridade, definida pela Procenge. Ao início de uma *Sprint*, essas requisições são discutidas e implementadas pela equipe de desenvolvimento. No momento da entrega, os detalhes da solução são documentados para uma posterior consulta de outros clientes e desenvolvedores.

A cada dois meses, um documento é redigido com todos os indicadores de qualidade e produtividade para ser mostrado aos gestores da Procenge. Existem também documentos com instruções relativas à manutenção da qualidade de todo o processo, que todos os membros da equipe devem seguir.

O apoio do Scrum, junto à prática das metodologias tradicionais, proporcionou, segundo os profissionais de qualidade e de desenvolvimento, maiores índices de satisfação dos clientes, melhores índices de produtividade das equipes e maior qualidade ao software. Tornando-o, portanto, mais confiável.

Capítulo 5

Conclusões e trabalhos futuros

A partir do estudo feito, conclui-se que o uso das metodologias tradicionais está cedendo seu espaço, pelo menos nas áreas de manutenção de sistemas, às metodologias ágeis devido à sua alta burocracia, que exige todo o planejamento já na fase inicial do projeto. Já a metodologia ágil, chega para tornar o ambiente de desenvolvimento de softwares mais produtivo e dinâmico, ajudando às empresas a conquistarem mais clientes e trazendo mais qualidade aos seus produtos.

Nas metodologias ágeis, os profissionais tornam-se mais especializados, ao passo que eles precisam ter conhecimento de análise, programação e testes. Com isso, as empresas encontram dificuldades em contratar pessoas com grandes níveis de conhecimento. É preciso também levar em conta que, dependendo do projeto, se torna necessária uma equipe relativamente grande para o desenvolvimento do sistema. No caso do Pirâmide, alguns módulos possuem mais profissionais que outros, devido ao seu uso ser mais intenso nos clientes.

Com o uso de metodologias tradicionais, no final de um projeto, o sistema pode ficar defasado em relação aos requisitos do cliente, caso as necessidades do negócio do cliente mudem com o tempo. Nas metodologias ágeis, o desenvolvimento do sistema acompanha as atualizações nos processos da empresa solicitante.

As metodologias ágeis oferecem uma abordagem com foco maior na qualidade, diferentemente das metodologias tradicionais. As metodologias ágeis procuram sempre validar com o cliente quais são os requisitos mais importantes a serem desenvolvidos, garantindo maior qualidade e satisfação dos clientes.

No futuro, a ideia seria trabalhar algumas limitações das metodologias ágeis, como a falta de análise de riscos, que ainda é um ponto mais tratado pelas metodologias tradicionais. Sem essa análise mais profunda, um risco de grandes

proporções, que não foi visto anteriormente, pode aparecer já na fase de entrega do projeto, causando um atraso, que, por sua vez, pode causar insatisfação do cliente.

Ao trabalhar nesse ponto, as metodologias ágeis podem se tornar mais completas e auxiliar o uso de metodologias tradicionais nos projetos mais recentes, possibilitando o aparecimento de processos híbridos de desenvolvimento de sistemas.

Referências

[1] Agile Manifesto. Disponível em: http://www.agilemanifesto.org/iso/ptbr/. Acesso em: 20/10/2012.

[2] Brooks, F. No Silver Bullet: Essence and Accidents of Software Engineering. Proc. IFIP, IEEE CS Press, pp. 1069-1076; reprinted in IEEE Computer, pp. 10-19, Apr. 1987.

[3] Cockburn, A. e Highsmith, J. “Agile Software evelopment: The Business of Innovation”, IEEE Computer, Sept., pp. 120-122, 2001.

[4] Pressman, R. Engenharia de Software. McGraw-Hill, 6ª Edição. 2006

[5] Silva, F. L. de Carvalho e da Silva, V. B. “Estudo comparativo dos processos de desenvolvimento de software ágil e tradicional, sob a ótica do guia PMBOK”, Trabalho final do curso de pós-graduação em produção de sistemas no Instituto Federal de Educação, Ciência e Tecnologia Fluminense, 2009.

[6] Soares, S. M. “Comparação entre Metodologias Ágeis e Tradicionais para Desenvolvimento de Software”, INFOCOMP Journal of Computer Science, v. 3, n. 2, pp. 08-13, 2004.

[7] Sommerville, I. Engenharia de Software. Editora Addison-Wesley. 592p, 2003.

[8] Marina Martinez. RUP. Disponível em: http://www.infoescola.com/engenharia-de-software/rup/. Acesso em 15 de dez. 2012.

[9] Guia on-line do RUP. Disponível em: http://www.wthreex.com/rup/portugues/index.htm. Acesso em 15 de dez. 2012.

[10] Best Practices for Development Teams. Disponível em: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf. Acesso em: 15 de dez. 2012